

RRRRRRRR	MM	MM	11	GGGGGGGG	EEEEEEEEE	TTTTTTTT
RRRRRRRR	MM	MM	11	GGGGGGGG	EEEEEEEEE	TTTTTTTT
RR RR	RR	MMMM	MMMM	1111	GG	EE
RR RR	RR	MMMM	MMMM	1111	GG	EE
RR RR	RR	MM	MM	11	GG	EE
RR RR	RR	MM	MM	11	GG	EE
RRRRRRRR	MM	MM	11	GG	EE	TT
RRRRRRRR	MM	MM	11	GG	EE	TT
RR RR	MM	MM	11	GG GGGGGG	EE	TT
RR RR	MM	MM	11	GG GGGGGG	EE	TT
RR RR	MM	MM	11	GG GG	EE	TT
RR RR	MM	MM	11	GG GG	EE	TT
RR RR	MM	MM	111111	GGGGGG	EEEEEEEEE	TT
RR RR	MM	MM	111111	GGGGGG	EEEEEEEEE	TT

.....
.....
.....

LL		SSSSSSS
LL		SSSSSSS
LL		SS
LLLLLLLL		SSSSSSS
LLLLLLLL		SSSSSSS

RM1GET
Table of contents

SEQUENTIAL SPECIFIC GET

C 11

16-SEP-1984 00:47:30 VAX/VMS Macro V04-00

Page 0

(3) 84
(4) 113
(14) 401

DECLARATIONS
RMSGET1/RMSFIND1 - SEQ. \$GET & \$FIND
RMSSEQKEY - ROUTINE TO CONVERT REC NO. TO RFA FOR SEQ. ORG

0000 1 \$BEGIN RM1GET,000,RMSRMS1,<SEQUENTIAL SPECIFIC GET>,<NOWRT,QUAD>
0000 2
0000 3 *****
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26 :

0000 28 :++
0000 29 Facility: RMS32
0000 30
0000 31 Abstract:
0000 32 This module provides sequential file organization-
0000 33 specific processing for the \$GET and \$IND functions.
0000 34
0000 35 Environment:
0000 36 STAR processor running STARLET EXEC.
0000 37
0000 38 Author: L F Laverdure, CREATION DATE: 4-Feb-1977
0000 39
0000 40 Modified By:
0000 41
0000 42 V03-006 RAS0278 Ron Schaefer 24-Mar-1984
0000 43 Minor tweaking for performance/size.
0000 44
0000 45 V03-005 RAS0120 Ron Schaefer 25-Jan-1983
0000 46 Add support to echo SYSS\$INPUT to SYSS\$OUTPUT.
0000 47
0000 48 V03-004 KBT0414 Keith B. Thompson 30-Nov-1982
0000 49 Change ifb\$w_devbufsiz to ifb\$l_devbufsiz
0000 50
0000 51 V03-003 KBT0340 Keith B. Thompson 16-Sep-1982
0000 52 Correct some opps from kbt0086
0000 53
0000 54 V03-002 KBT0139 Keith B. Thompson 20-Aug-1982
0000 55 Reorganize psects
0000 56
0000 57 V03-001 KBT0086 Keith B. Thompson 13-Jul-1982
0000 58 Clean up psects
0000 59
0000 60 V02-022 RAS0063 Ron Schaefer 29-Jan-1982
0000 61 Correct probes of user key and record buffers.
0000 62
0000 63 V02-021 JWH0001 Jeffrey W. Horn 31-DEC-1981
0000 64 Fix broken branches.
0000 65
0000 66 V02-020 RAS0018 Ron Schaefer 7-Aug-1981
0000 67 Fix broken branch caused by stream files.
0000 68
0000 69 V02-019 REFORMAT Frederick E. Deen, Jr. 28-Jul-1980
0000 70 This code was reformatted to adhere to RMS standards
0000 71
0000 72 V018 CDS0001 C D Saether 11-Mar-1980
0000 73 Fix divide by 0 caused by record size > blocksize and
0000 74 no span attribute set (nonsense combination). Don't
0000 75 look at RAC in RAB to determine whether saved NRP context
0000 76 on find is to be restored.
0000 77
0000 78 V017 JAK0001 J A Krycka 27-Aug-1978
0000 79 Miscellaneous clean-up prior to DECNET V1.0 code freeze.
0000 80 Add code to support network access by key.
0000 81
0000 82 :--

0000 84 .SBTTL DECLARATIONS
0000 85
0000 86
0000 87 : INCLUDE FILES:
0000 88 :
0000 89 :
0000 90 :
0000 91 : MACROS:
0000 92 :
0000 93 :
0000 94 \$IRBDEF
0000 95 \$IFBDEF
0000 96 \$RABDEF
0000 97 \$FABDEF
0000 98 \$DEVDEF
0000 99 \$RMSDEF
0000 100
0000 101
0000 102 : EQUATED SYMBOLS:
0000 103 :
0000 104
0000 105 BKP=IRB\$L_BKPBITS*8
0000 106 ROP=RAB\$L_ROP*8 : bit offset to ROP
0000 107
0000 108 :
0000 109 : OWN STORAGE:
0000 110 :
0000 111

0000 113 .SBTTL RMSGET1/RMSFIND1 - SEQ. \$GET & \$FIND
0000 114
0000 115 ++
0000 116 RMSGET1/RMSFIND1 - Common sequential file organization for \$GET and \$FIND
0000 117
0000 118 This module performs the following functions:
0000 119
0000 120 1. Common \$GET/\$FIND setup
0000 121 2. For UNIT RECORD DEVICES calls RMSGET_UNIT_REC
0000 122 otherwise, performs additional setup and calls RMSGET_BLK_DEV
0000 123
0000 124
0000 125
0000 126
0000 127 CALLING SEQUENCE:
0000 128 Entered via CASE branch from RMSSGET
0000 129 or RMSSFIND at RMSGET1 or RMSFIND1 respectively.
0000 130
0000 131 EXIT is to user via RMSEXRMS.
0000 132
0000 133
0000 134 INPUT PARAMETERS:
0000 135
0000 136 R11 IMPURE AREA address
0000 137 R10 IFAB address
0000 138 R9 IRAB address
0000 139 R8 RAB address
0000 140
0000 141
0000 142 IMPLICIT INPUTS:
0000 143
0000 144 The contents of the RAB and RELATED IRAB and IFAB.
0000 145 In particular, IRBSV_FIND must be set if doing \$FIND, else clear.
0000 146
0000 147 OUTPUT PARAMETERS:
0000 148
0000 149 R7 - R1 Destroyed
0000 150 R0 STATUS
0000 151
0000 152
0000 153 IMPLICIT OUTPUTS:
0000 154
0000 155 Various fields of the RAB are filled in to reflect the status of
0000 156 the operation (see functional spec for details).
0000 157
0000 158 The IRAB is similarly updated.
0000 159
0000 160
0000 161 COMPLETION CODES:
0000 162 STANDARD RMS (see functional spec).
0000 163
0000 164
0000 165 SIDE EFFECTS:
0000 166
0000 167 None
0000 168
0000 169 --

0000 171 RMSGET1::
22 A8 B4 0000 172 STSTPT GET1
2A AA B5 0009 173 CLRW RABSW_RSZ(R8) ; init the RSZ field
10 13 000C 174 TSTW IFBSW_ECHO_ISI(R10) ; any ISI value?
OC 6A 2E E1 000E 175 BEQL RMSFIND1 ; no echo if not
08 69 22 E1 0012 176 BBC #IFBSV_PPF_INPUT,(R10),RMSFIND1 ; not if not sysin
0016 177 BBC #IRBSV_PPF_IMAGE,(R9),RMSFIND1 ; not if not indirect
001A 178 CSB #IRBSV_ASYNC,(R9) ; force synchronous
001E 179 SSB #IRBSV_PPF_ECHO,(R9) ; ask for echo
001E 180
001E 181 : Make user input valid checks
001E 182 :
001E 183 :
001E 184 :
001E 185 RMSFIND1::
62 A9 B4 001E 186 STSTPT FIND1
0024 187 CLRW IRBSW_CSIZ(R9) ; clear current record size
0027 188 ; indicates no current record
0027 189 :
0027 190 : Check for UNIT RECORD DEVICE and if so dispatch to appropriate code
0027 191 :
0027 192 :
33 6A E8 0027 193 ASSUME DEV\$V_REC EQ 0
0027 194 BLBS IFBSL_PRIM_DEV(R10),GET_UR ; branch if UNIT RECORD
002A 195
002A 196 :
002A 197 : Files oriented device - dispatch on record access type
002A 198 :
002A 199 :
002A 200 CASE TYPE=B, SRC=RABSB_RAC(R8),-
002A 201 DISPLIST=<GETSEQ,GETKEY,GETRFA>

0035 203 :
0035 204 : Handle errors
0035 205 :
0035 206 :
2D 11 003A 207 ERRRAC: RMSERR RAC
003C 208 BRB G_XIT
46 18 6A E0 003C 210 ERRRFA: BBS #DEV\$V FOR,-
003E 211 IFBSL_PRIM_DEV(R10),RFAOFF ; allow LBN 0 for NFS operation
0040 212 RMSERR RFA
22 11 0045 213 BRB G_XIT
0047 214 :
1B 11 0047 215 ERRIOP: RMSERR IOP
004C 216 BRB G_XIT
004E 217 :
04 2E 6A 22 E1 004E 218 EPRSQQ: BBC #IFBSV_PPF_INPUT,(R10),10\$
2B 69 22 E1 0052 219 BBC #IRBSV_PPF_IMAGE,(R9),TSTRFA ; branch if not 'INPUT'
0056 220 10\$: RMSERR SQ0 ; continue if not image
OC 11 005B 221 BRB G_XIT ; not SEQ. operation

```

005D 223 ;++
005D 224
005D 225 ; GET for UNIT RECORD DEVICE
005D 226
005D 227 ;--
005D 228
05 6A 3E E0 005D 229 ASSUME RABSC SEQ EQ 0
1E A8 95 0061 230 GET_UR: BBS #IFBSV DAP,(R10),10$ ; branch if network operation
CF 12 0064 231 TSTB RABSB_RAC(R8) ; allow sequential access only
FF97* 30 0066 232 BNEQ ERRRAC ; branch on error
6C 11 0069 233 10$: BSBW RMSGET_UNIT_REC ; perform UNIT RECORD GET
006B 234 G_XIT: BRB GETXIT ; and branch to GET EXIT
006B 235
006B 236 ; Keyed access type
006B 237
006B 238
0088 30 006B 239 GETKEY: BSBW RM$SEQKEY ; convert key to RFA
66 50 E9 006E 240 BLBC R0,GETXIT ; get out on error
0071 241
0071 242
0071 243 ; GET by RFA
0071 244
0071 245
0071 246
04 69 29 E1 0071 247 GETRFA:
0071 248 BBC #IRBSV FIND, (R9), 10$ ; branch if doing $GET
7E 40 A9 7D 0075 249 ASSUME IRBSW_NRP_OFF EQ IRBSL_NRP_VBN+4 ; save NRP context for random
0075 250 MOVO IRBSL_NRP_VBN(R9), -(SP) ; FIND keeping stack long aligned
0079 251
0079 252
0079 253 ; Check for valid RFA value
0079 254
0079 255
0079 256
(CD 6A 1C E1 0079 257 10$: BBC #DEVSV RND,- ; branch if no random access
CA 6A 2D E0 007B 258 IFBSL_PRIM_DEV(R10),ERRIOP
10 A8 D5 007D 259 TSTRFA: TSTL #IFBSV SQO,(R10),ERRSQO ; branch if SQO specified
B6 13 0081 260 BEQL RABSW_RFA(R8) ; zero RFA?
14 A8 B1 0084 261 RFAOFF: CMPW ERRLRFA ; branch if yes
48 AA 0086 262 IFBSL_RFA+4(R8),- ; offset within a block?
AF 1E 0089 263 BGEQU ERRLRFA ; branch if not
40 A9 10 A8 7D 008D 264 ASSUME IRBSW_NRP_OFF EQ IRBSL_NRP_VBN+4 ; copy RFA to NRP
46 A9 B4 0092 265 MOVO RABSW_RFA(R8),IRBSL_NRP_VBN(R9) ; guarantee offset valid long word
0095 266 CLRW IRBSL_NRP_OFFSET+2(R9)
0095 267
0095 268
0095 269 ; Do final setups for random access GET.
0095 270 ; Clear LAST FIND and EOF status bits.
0095 271
0095 272
0095 273
04 A9 22 8A 0095 274 BICB2 #<1@<IRBSV FIND LAST-BKP>>!- ; branch if file has last block
0099 275 <1@<IRBSV_EOF-BKP>>,IRBSL_BKPBITS(R9)
0099 276
0099 277 ; Determine number of blocks to read in based upon the longest record
0099 278
0099 279 ; in the file.

```

			0099	280 :			
			0099	281			
			009C	282	ADDW3	IRBSW_NRP OFF(R9),-	
			009F	283	CMPB	IFBSW_LRLTR10), R2	
			00A3	284	BEQL	IFBSB_RFMOORG(R10),#FABSC_FIX	
			00A5	285	10\$:	10\$	
			00A8	286	ADDW2	#2,R2	
			00AA	287	DECW	R2	
			00AF	288	EXTZV	#9,#7,R2,R2	
			00B3	289	CMPB	R2,IRBSB_MBC(R9)	
			00B5	290	BGEQU	30\$	
			00B8	291	MOVL	#2 R3	
			00BF	292	BSBW	RM\$GETRANDOM	
			00C3	293	20\$:	#IRBSV FIND, (R9), GETXIT	
			00C5	294	BBC	IRBSW_NRP OFF EQ IRBSL_NRP_VBN+4	
			00C8	295	ASSUME	(SP)+, IRBSL_NRP_VBN(R9)	
				296	MOVO	BRB	GETXIT
				297	30\$:	BSBW	RMSGET_BLK_DEV
				299		BRB	20\$

; compute total # bytes needed
 ; fixed length records?
 ; branch if yes
 ; add in size field overhead
 ; round down
 ; get # blks - 1
 ; < MBC?
 ; branch if not
 ; set flag for short read
 ; go read and return the record
 ; all done if this was a GET
 ; clean saved NRP info off STACK
 ; all done
 ; GET the record
 ; return to mainline

00CA 301
00CA 302 :++
00CA 303
00CA 304 Sequential access SGET
00CA 305
00CA 306 :--
00CA 307
00CA 308 .ALIGN LONG
04 69 29 E0 00CC 309
11 69 25 E0 00D0 310 BBS #IRB\$V_FIND,(R9),GETSQ1
FF29' 30 00D4 311 BBS #IRB\$V_FIND_LAST,(R9),RESET_NRP ; branch if doing \$FIND
00D7 312 GETSQ1: BSBW RMSGET_BLK_DEV ; branch if \$FIND was last
00D7 313 ; go get the record
00D7 314 :++
00D7 315 : Exit GET function. Set LAST-OPERATION-WAS-A-FIND status as required.
00D7 316
00D7 317
00D7 318 :--
00D7 319
11 69 29 E4 00D7 320 GETXIT: CSB #IRB\$V_FIND_LAST,(R9) ; clear LAST-OPERATION WAS A FIND
2F 50 E9 00DB 321 BBSC #IRB\$V_FIND,(R9),FNDXIT ; branch if doing a FIND
FF1B' 31 00DF 322 CHKERR: BLBC R0,GETERR ; branch on error
00E2 323 EXIT: BRW RM\$EXRMS ; R0 still has SUC/ERR code
00E5 324
00E5 325 :++
00E5 326 : This is a SGET operation after a SFIND. Reset NRP to RP.
00E5 327
00E5 328
00E5 329 :--
00E5 330
00E5 331 ASSUME IRBSW_RP OFF EQ IRBSL_RP VBN+4
00E5 332 ASSUME IRBSW_NRP_OFF EQ IRBSL_NRP_VBN+4
00E5 333 RESET_NRP:
48 A9 7D 00E5 334 MOVQ IRBSL_RP VBN(R9) - ; reset NRP
40 A9 21 E5 00E8 335 BBCC #IRB\$V_EOF,(R9),GETSQ1 ; make sure EOF flag off
E6 69 11 00EA 336 BRB GETSQ1 ; rejoin GET code
E4 11 00EE 337

R
M1GET
V04-000

SEQUENTIAL SPECIFIC GET
RM1GET1/RMSFIND1 - SEQ. SGET & SFIND

M 11

16-SEP-1984 00:47:30 VAX/VMS Macro V04-00
5-SEP-1984 16:23:18 [RMS.SRC]RM1GET.MAR;1

Page 10
(11)

00F0 339
00F0 340 :++
00F0 341
00F0 342 : Exiting from a SFIND. Check for PROCESS-PERMANENT FILE special processing
00F0 343 : and setting of the FIND_LAST flag.
00F0 344 :--
00F0 345
00F0 346
00F0 347 FNDXIT: BLBC R0_GETERR
15 69 2E 50 E9 00F0 348 #IRBSV_PPF_EOF,(R9) PPF EOF
06 69 2F E4 00F3 349 #IRBSV_PPF_SKIP,(R9) PPF SKIP
E0 69 25 E4 00F7 350 #IRBSV_FIND_LAST,(R9),CHRERR
DE 11 00FF 351 BRB CHKERR
; branch if operation failed
; branch if SYSSINPUT EOF
; or must skip record
; set last opr. was a FIND
; rejoin GET code

0101 353
 0101 354 :++
 0101 355
 0101 356 : IRBSV_PPF_SKIP was set (now clear).
 0101 357 This indicates that we have just skipped over a \$DECK record on SYSSINPUT.
 0101 358
 0101 359 Restore IRBSV_FIND from IRBSV_PPF_FNDSV and redo the SFIND or \$GET.
 0101 360
 0101 361
 0101 362 :--
 0101 363 PPF_SKIP:
 04 69 30 E5 0101 364 BBCC #IRBSV_PPF_FNDSV,(R9),10\$: branch if not doing SFIND
 FF12 31 0105 365 SSB #IRBSV_FIND,(R9) : set FIND bit
 0109 366 10\$: BRW RMSFIND1 : redo SFIND or \$GET
 010C 367
 010C 368 :++
 010C 369
 010C 370 : IRBSV_PPF_EOF was set (now clear).
 010C 371 This indicates that either a \$EOD or user-defined EOD-STRING was seen
 010C 372 and has been skipped.
 010C 373
 010C 374 Change status code to RMSS_EOF
 010C 375
 010C 376 :--
 010C 377
 010C 378 PPF_EOF:
 010C 379 RMSEERR EOF
 0111 380
 0111 381 :++
 0111 382
 0111 383 : An error has occurred.
 0111 384
 0111 385 : Unless error is 'RTB', zero current record size.
 0111 386
 0111 387 :--
 0111 388
 0111 389 GETERR:
 81A8 8F 50 B1 0111 390 CMPW R0,#RMSS_RTBB^XFFFF : was error RTB?
 08 13 0116 391 BEQL EXIT1 : branch if yes
 62 A9 84 0118 392 CLRW IRBSW CSIZ(R9) : invalidate current rec indicator
 CA 0118 393 BICL2 #<1a<IRBSV_PPF_EOF-BKP>>!-
 011C 394 <1a<IRBSV_PPF_SKIP-BKP>>!-
 011C 395 <1a<IRBSV_PPF_ECHO-BKP>>!-
 011C 396 <1a<IRBSV_PPF_FNDSV-BKP>>,-
 04 A9 0201C000 8F FEDA' 31 0123 397 IRBSL_BKPBITSTR9)
 FEDA' 31 0123 398 EXIT1: BRW RMSEXRMS
 0126 399

0126 401 .SBTTL RMSSEQKEY - ROUTINE TO CONVERT REC NO. TO RFA FOR SEQ. ORG
 0126 402
 0126 403 ++
 0126 404 RMSSEQKEY - Convert rec nbr. to RFA for sequential organization
 0126 405
 0126 406 This routine converts a record number to an RFA for fixed length
 0126 407 records.
 0126 408
 0126 409
 0126 410
 0126 411
 0126 412
 0126 413
 0126 414
 0126 415 R10 IFAB address
 0126 416 R9 IRAB address
 0126 417 R8 RAB address
 0126 418 RABSL_KBF Address of buffer having the relative record
 0126 419 number (RRN)
 0126 420 RABSB_KSZ Size of key (must be 4 - defaulted if 0)
 0126 421
 0126 422
 0126 423
 0126 424
 0126 425 IFBSV_BLK Set if records cross block boundaries
 0126 426 IFBSW_MRS Fixed record length
 0126 427
 0126 428
 0126 429
 0126 430 RABSW_RFA Set to VBN and offset in block for record
 0126 431 R0 STATUS code
 0126 432 R1-R3 Destroyed
 0126 433
 0126 434
 0126 435
 0126 436 Standard RMS, in particular, RAC, KBF, KSZ, KEY, IOP, or SUC.
 0126 437
 0126 438
 0126 439
 0126 440
 0126 441
 0126 442
 0126 443
 0126 444 RMSSEQKEY:::
 01 50 AA 91 0126 445 CMPB IFBSB RFMORG(R10),#FABSC_FIX : must be fixed RFM
 5B 12 012A 446 BNEQ ERRRACT1 : too bad if not
 34 AB 95 012C 447 TSTB RABSB_KSZ(R8) : zero size = 4
 06 13 012F 448 BEQL 10\$: default buffer size
 04 34 AB 91 0131 449 CMPB RABSB_KSZ(R8),#4 : if not default must be 4
 5C 12 0135 450 BNEQ ERRKSZ
 0137 451
 0137 452 : Pick up record number getting rid of bias (i.e., there is no record 0)
 0137 453
 0137 454
 0137 455
 50 30 B8 DE 0137 456 10\$: MOVAL ARABSL_KBF(R8),R0 : key buffer addr
 0138 457 IFNORD #4,(ROT,-) : check access

51	60	01	C3	0138	458		ERRKBF	IRBSB_MODE(R9)		
		51	19	0142	459	SUBL3	#1,(R0),R1			
				0146	460	BLSS	ERRKEY			
				0148	461					
				0148	462					
				0148	463					
				0148	464					
				0148	465					
50	60	AA	3C	0148	466	MOVZWL	IFBSH_MRS(R10),R0			
	50		D6	014C	467	INCL	R0			
	50	01	CA	014E	468	BICL2	#1,R0			
7E	0200	8F	3C	0151	469	MOVZWL	#512-(SP)			
		03	E0	0156	470	BBS	#FAB\$V_BLK,-			
	10	51	AA	0158	471		IFBSB_RAT(R10),BLKSET			
				015B	472					
				015B	473					
				015B	474					
				015B	475					
				015B	476					
				015B	477					
				015B	478					
52	6E	51	50	7A	015B	479	EMUL	R0,R1,(SP),R2		
14 A8	10 A8	52	8E	7B	0160	480				
				0160	481	EDIV	(SP)+,R2,-			
				0167	482		RABSW_RFA(R8),RABSW_RFA+4(R8)			
				0167	483	SUCRET:	RMSSUC			
		05	016A	484		RSB				

```

016B 486
016B 487
016B 488 ; Alternate calculation for records not allowed to cross block boundaries
016B 489
016B 490
53 8E 52 D4 016B 491 BLKSET: CLRL R2
      50 C7 016D 492 DIVL3 R0,(SP)+,R3
      11 13 0171 493 BEQL ERRIOP_BR
      0173 494
      0173 495
      0173 496
      0173 497
      0173 498
52 51 51 53 7B 0173 499 EDIV R3,R1,R1,R2
      0178 500
      0178 501 ADDL3 #1,R1,RABSW_RFA(R8)
      10 A8 51 01 0178 502 MULW3 R2,R0,RABSW_RFA+4(R8)
      14 A8 50 52 A5 017D 503
      E3 11 0182 504 BRB SUCRET
      0184 505
      0184 506
      0184 507 ; Handle errors
      0184 508
      0184 509
      0184 510
      0184 511 ERRIOP_BR:
FEC0 31 0184 512 BRW ERRIOP ; extended BRANCH
      0187 513 ERRRAC1: RMSERR RAC ; not fixed length records
      05 018C 514 RSB
      018D 515
      018D 516 ERRKBF: RMSERR KBF ; invalid KEY buffer addr
      05 0192 517 RSB
      0193 518
      0193 519 ERRKSZ: RMSERR KSZ ; bad KEY size (not 0 or 4)
      05 0198 520 RSB
      0199 521
      0199 522 ERRKEY: BBC #DEV$V FOR,-
      09 6A 0199 523 IFBSL_PRIM_DEV(R10),10$ ; definite error if not for.
      60 D5 019D 524 TSTL (R0) ; attempt to read LBN 0?
      05 12 019F 525 BNEQ 10$ ; branch if not (error)
      10 A8 7C 01A1 526 CLRQ RABSW_RFA(R8) ; yes, zero LBN
      C1 11 01A4 527 BRB SUCRET ; continue
      01A6 528
      01A6 529 10$: RMSERR KEY ; bad KEY value (not > 0)
      05 01AB 530 RSB
      01AC 531
      01AC 532
      01AC 533 .END

```

\$\$PSECT EP	= 00000000	IRBSV_PPF_SKIP	= 0000002F
\$\$RMSTEST	= 0000001A	IRBSW_CSIZ	= 00000062
\$\$RMS_PBUGCHK	= 00000010	IRBSW_NRP OFF	= 00000044
\$\$RMS_TBUGCHK	= 00000008	IRBSW_RP OFF	= 0000004C
\$\$RMS_UMODE	= 00000004	PIOSA_TRACE	***** X 01
BKP	= 00000020	PPF_EOF	0000010C R 01
BLKSET	0000016B R 01	PPF_SKIP	00000101 R 01
CHKERR	000000DF R 01	RABSB_KSZ	= 00000034
DEVSF_FOR	= 00000018	RABSB_RAC	= 0000001E
DEVSF_REC	= 00000000	RABSC_SEQ	= 00000000
DEVSF_RND	= 0000001C	RABSL_KBF	= 00000030
ERRIOP	00000047 R 01	RABSL_ROP	= 00000004
ERRIOP_BR	00000184 R 01	RABSW_RFA	= 00000010
ERRKBF	0000018D R 01	RABSW_RSZ	= 00000022
ERRKEY	00000199 R 01	RESET_NRP	000000E5 R 01
ERRKSZ	00000193 R 01	RFAOFF	00000086 R 01
ERRRAC	00000035 R 01	RMSEXRMS	***** X 01
ERRRAC1	00000187 R 01	RMSFIN1	0000001E RG 01
ERRRFA	0000003C R 01	RMSGT1	00000000 RG 01
ERRSQQ	0000004E R 01	RMSGETRANDOM	***** X 01
EXIT	000000E2 R 01	RMSGET_BLK_DEV	***** X 01
EXIT1	00000123 R 01	RMSGET_UNIT_REC	***** X 01
FABSC_FIX	= 00000001	RMSS_EQKEY	00000126 RG 01
FABSV_BLK	= 00000003	RMSS_EOF	= 0001827A
FNDXIT	000000F0 R 01	RMSS_IOP	= 00018574
GETERR	00000111 R 01	RMSS_KBF	= 0001858C
GETKEY	0000006B R 01	RMSS_KEY	= 00018594
GETRFA	00000071 R 01	RMSS_KSZ	= 000185A4
GETSEQ	000000CC R 01	RMSS_RAC	= 00018644
GETSQ1	000000D4 R 01	RMSS_RFA	= 0001865C
GETXIT	000000D7 R 01	RMSS_RTB	= 000181A8
GET_UR	0000005D R 01	RMSS_SQO	= 000186C4
G_XIT	00000069 R 01	ROP	= 00000020
IFBSB_RAT	= 00000051	SUCRET	00000167 R 01
IFBSB_RFMOrg	= 00000050	TPTSL_FIND1	***** X 01
IFBSL_DEVBUFSIZ	= 00000048	TPTSL_GET1	***** X 01
IFBSL_PRIM_DEV	= 00000000	TSTRFA	00000081 R 01
IFBSV_DAP	= 0000003E		
IFBSV_PPF_INPUT	= 0000002E		
IFBSV_SQO	= 0000002D		
IFBSW_ECHO_ISI	= 0000002A		
IFBSW_LRL	= 00000052		
IFBSW_MRS	= 00000060		
IRBSB_MBC	= 00000055		
IRBSB_MODE	= 0000000A		
IRBSL_BKPBITS	= 00000004		
IRBSL_NRP_OFF	= 00000044		
IRBSL_NRP_VBN	= 00000040		
IRBSL_RP_VBN	= 00000048		
IRBSV_ASNC	= 00000023		
IRBSV_EOF	= 00000021		
IRBSV_FIND	= 00000029		
IRBSV_FIND_LAST	= 00000025		
IRBSV_PPF_ECHO	= 00000039		
IRBSV_PPF_EOF	= 0000002E		
IRBSV_PPF_FNDSV	= 00000030		
IRBSV_PPF_IMAGE	= 00000022		

RM1GET
Psect synopsis

SEQUENTIAL SPECIFIC GET

F 12

16-SEP-1984 00:47:30 VAX/VMS Macro V04-00
5-SEP-1984 16:23:18 [RMS.SRC]RM1GET.MAR;1

Page 16
(16)

+-----+
! Psect synopsis !
+-----+

PSECT name

	Allocation	PSECT No.	Attributes																	
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE							
RMSRMS1	0000001AC (428.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	QUAD							
\$ABSS	00000000 (0.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							

+-----+
! Performance indicators !
+-----+

Phase

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.05
Command processing	107	00:00:00.67	00:00:04.39
Pass 1	303	00:00:09.63	00:00:27.97
Symbol table sort	0	00:00:01.24	00:00:01.67
Pass 2	106	00:00:02.17	00:00:05.34
Symbol table output	12	00:00:00.10	00:00:00.46
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	561	00:00:13.89	00:00:40.90

The working set limit was 1350 pages.

53909 bytes (106 pages) of virtual memory were used to buffer the intermediate code.

There were 50 pages of symbol table space allocated to hold 998 non-local and 15 local symbols.

533 source lines were read in Pass 1, producing 14 object records in Pass 2.

24 pages of virtual memory were used to define 23 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	12
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	19

1105 GETs were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:\$RM1GET/OBJ=OBJ\$:\$RM1GET MSRC\$:\$RM1GET/UPDATE=(ENH\$:\$RM1GET)+EXECMLS/LIB+LIB\$:\$RMS/LIB

0321 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

